SIAT - Sistema de Irrigação Automatizado

Ana Laura Novaes
Karoline Gomes Cipriano
Leticia Eleutério Alves
Rosália Pricila Guimarães
Sabrina de Fátima Morandini

Agosto de 2014/Novembro de 2014

1 índice

1. Introdução 3

2. Descrição 4

3. Objetivo 4

4. Materiais Utilizados 5

5. Desenvolvimento 6
   Hardware 6
   Software 8

6. Resultados obtidos 9

7. Conclusões 11

8. Referências Bibliográficas 12

9. Apêndices 13

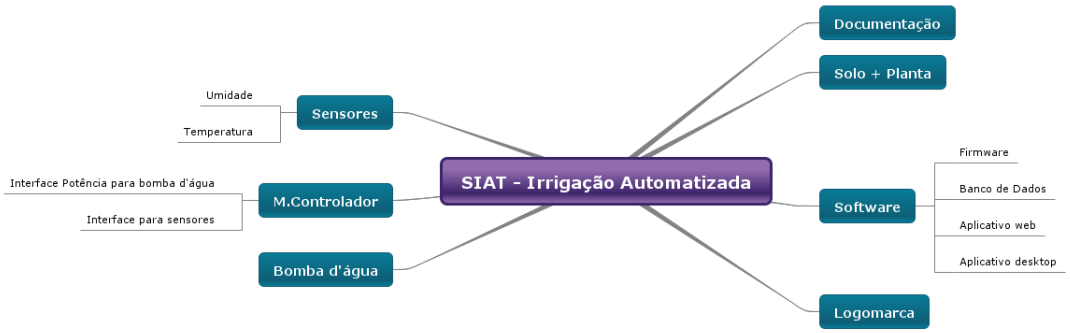
2 Introdução

A automação de sistemas de irrigação vem sendo implantada com maior intensidade nos últimos anos, principalmente em função do surgimento de técnicas apropriadas que vem acompanhando a modernização crescente da agricultura e abertura do mercado brasileiro às importações, principalmente com relação à irrigação localizada, liderada por empresas americanas, israelenses e européias.
A automação se faz necessária não somente pela possibilidade de diminuição dos custos com mão de obra, mas principalmente por necessidades operacionais, tais como irrigação de grandes área no período noturno.
Pode-se destacar como vantagens da automação de sistemas de irrigação os seguintes itens:

- Possibilita irrigações noturnas sem necessidade de acompanhamento;
- Diminui a potência de acionamento;
- Diminui custo de bombeamento;
- Precisão nos tempos e turnos de irrigação;
- Eficiência na aplicação de água;
- Jardins mais belos e saudáveis;
- Economia de tempo, água e mão-de-obra;
- Protege os jardins de pragas e doenças;
- Adubação líquida com melhor resultado;
- Controle automático de água.

3 Descrição

O projeto SIAT foi desenvolvido com a finalidade de poupar atividades manuais, para poder implementar a automatização em processos comumentes utilizados para o sistema de irrigação do solo, em geral.



4 Objetivo

Montar o projeto de acordo com os planos estipulados pelo grupo, dando a atenção necessária a cada detalhe para que os resultados atingidos sejam satisfatórios.
Além de se obter um maior conhecimento pela área do curso técnico de Automação, focando em softwares e hardwares especificos para criação de projetos como este apresentado.

5 Materiais Utilizados

- Protoboard;
- Trimpot;
- Multímetro;
- Resistor;
- Maquete do solo (Areia);
- Sensor de umidade;
- Sensor de nível;
- PIC18F2550;
- Arduino Uno;
- Regulador para 5 volts;
- Led;
- Tubo de Latex;
- Eletrobomba de Lavagem de 12V;

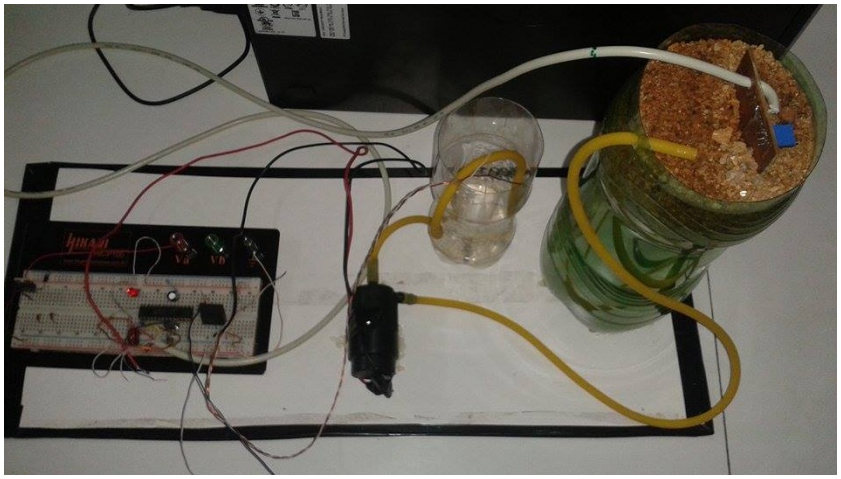
6 Desenvolvimento

Começou-se a realizar a montagem do hardware, através de maquetes e circuitos, e a montagem do software, através de programas previamentes instalados no computador e também de programas obtidos pela internet.

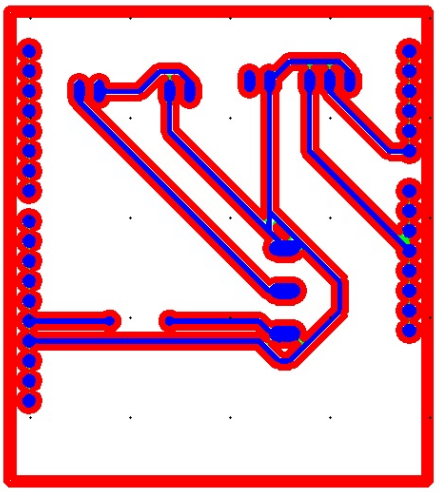
6.1 Hardware

Primeiramente, utilizando o sensor de umidade, mediu-se a resistência do solo, onde RS é aproximadamente igual a R:
*28K77 ohms - Solo seco;
*23K12 ohms - Solo umido;
*18K88 ohms - Solo encharcado.
Para que pudessemos calcular a tensão inicial no sensor seco e também molhado sem que este fosse ligado a uma fonte de energia.
Tensão: 5V
R: 23K22 ohms
Rs seco: 5,2V
Rs umido: 3,8V
Rs encharcado: 1,9V
Ao fazer as medições, ocorreu alguns problemas com o trimpote e observou-se que ao aumentar a área da placa, a resistência do solo(Rs) diminui.
Outros problemas ocorreram, só que desta vez, com o resistor R pois não se tinha o valor disponível para efetuar a montagem do hardware com a areia, já que ela não escoava a água rapidamente implicando em uma demora nas medições. Por isso, se utilizou de um resistor com o valor de 18K ohms e trocou-se a areia utilizada por uma mais fina para só então assim poder efetuar todas as medidas. Onde:
*R: 39K ohms (divisor de tensão)
*Rs seco: 1,15M ohms
*Rs umido: 0,37M ohms
*Rs encharcado: 41,3K ohms
Para que a bomba não ligue se não houver água no reservatório, um sensor de nível foi projetado. Este informará para o PIC se o nivel está baixo ou alto, ou seja, se o reservatório precisa ou não ser abastecido com a água.
Após realizar algumas configurações no PIC foi realizada a montagem em uma plataforma. Após a montagem foi feito um teste com a bomba para ver se o PIC estava funcionando.

Verificou-se um erro no PIC pois quando o solo estava seco, ele irrigava, porém não havia um limite de água a ser irrigada. O PIC estava lendo o valor atual da resistência mas não estava limitando-a de acordo com a resistencia Rs já estabelecida devido a um erro de endereçamento. Devido a esta falha e outras complicaçoes no projeto, optou-se por voltar a utilizar o Arduino e todas as medições foram refeitas.
Solo seco: 15M ohm
Solo seco: 3,9 V
Solo seco com sensor: 4,0V
R água: 1M ohm
A0: sensor de umidade
A1: nivel de água do reservatório;
Saída digital 13 aciona a bomba d'água;

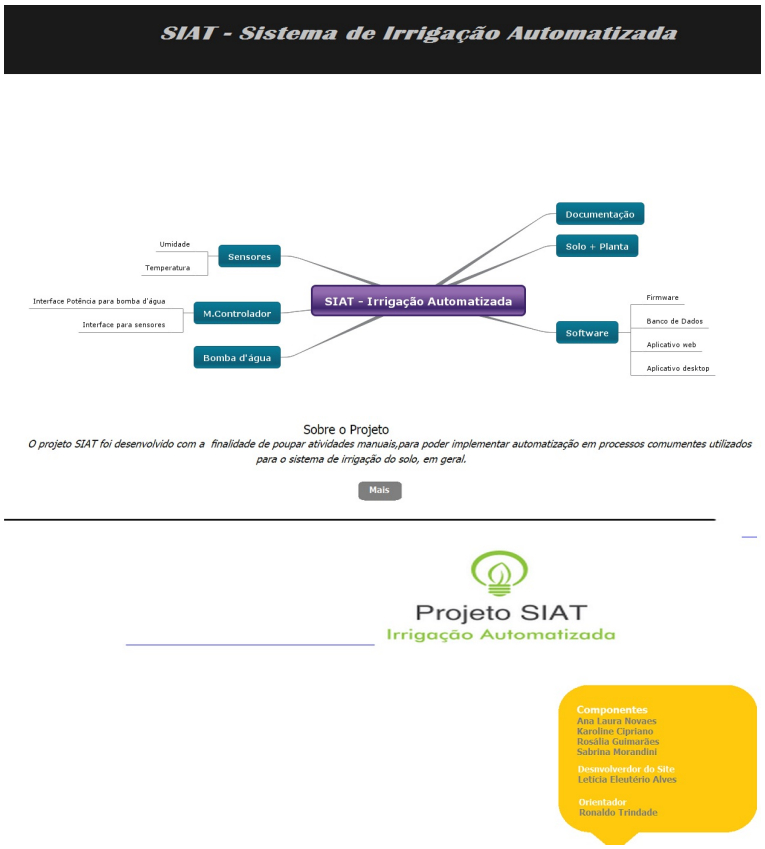


A figura acima representa o projeto em fase de acabamento, isto é, faltando apenas a placa de circuito impresso para a montagem final e entrega do mesmo. A desenho da placa do circuito impresso é mostrada abaixo:



6.2 Software

Primeiramente, aprendeu-se a utilizar o programa Codeigniter para poder mexer no HTML. Usou-se também o Filezilla e Firebug para conectar o ambiente virtual com o computador.
Para fazer a interface do site, utilizou-se o Kompozer, e o Arduino para fazer o codigo do próprio Arduino. A interface do site foi planejada em um rascunho no próprio Kompozer, assim como o logotipo do projeto, que será utilizado para a identificação e no design do site.
De início o site apresentou alguns problemas quanto ao código, mas que estavam sendo resolvidos. A interface inicial dele é apresentada abaixo:



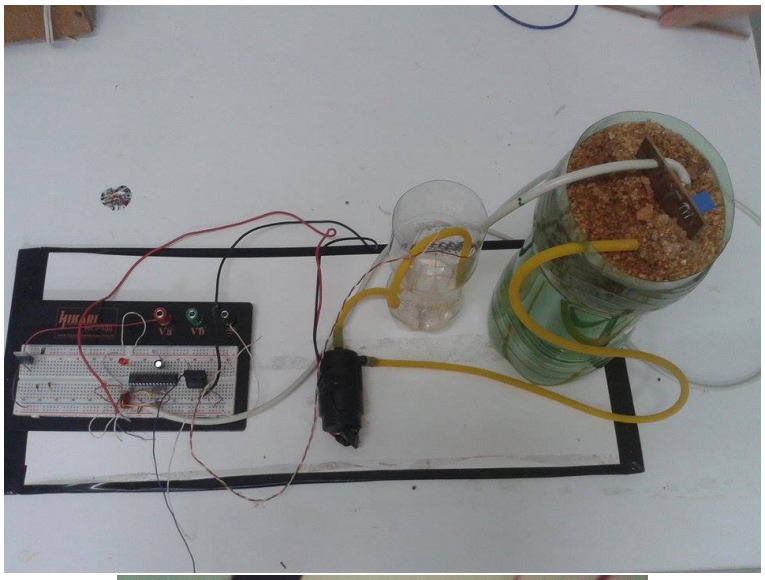
Testou-se também o arduino no circuito para ver quais as voltagens com o resistor R (R:39K ohms) define o Rs como seco, úmido ou encharcado.
Algumas ideias foram criadas e o banco de dados foi proposto. Ele irá fazer a ponte entre o website e o microcontrolador. Também foi criado duas tabelas para esse Banco de Dados, sendo uma para que o usuário informe se a irrigação está "ligada"ou "desligada", e uma outra para que o usuário informe por quanto tempo deseja que a irrigação seja feita.
É importante resaltar que a função do website é de apresentar o projeto em geral, ou seja, é um site de visitação.

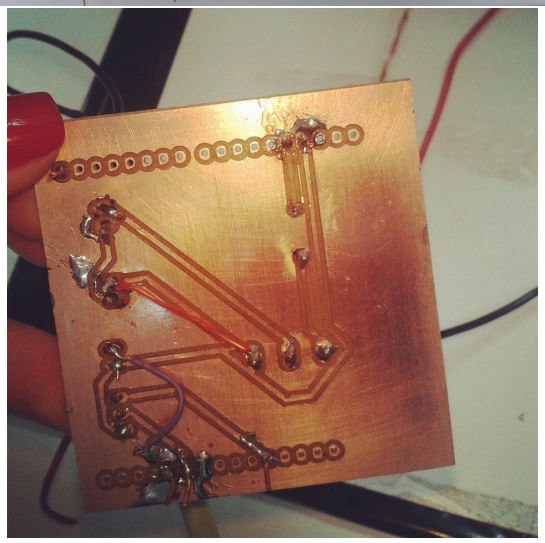
7 Resultados Obtidos

Devido a problemas de códigos fonte, a interface do website foi alterada. De modo a ficar cada vez mais acessível e compreendida pelos componentes do grupo do projeto e bem como o próprio usuário do website.

Sistema de Irrigação Automatizada
Instituto Federal Minas Gerais
Campus Ouro Preto
CODAAUT
Orientador: Ronaldo Trindade
SIAT
Irrigação Automatizada
O sistema de irrigação automatizado baseado na umidade do solo é um projeto que tem por objetivo o fornecimento controlado de água para as plantas.
Mais >>
Componentes: Ana Laura Novaes
Karoline Cipriano
Rosália Guimarães
Sabrina Morandini
Desenvolvedor: Letícia Alves
Sensores
Arduino
Resistência
Hardware | Software
Desenvolvimento da web site através da estrutura do Codeigniter.
O sistema consiste em um circuito composto pelos seguintes módulos: uma fonte de alimentação alternada para a alimentação dos sensores, um regulador de tensão, um sensor de solo que ira medir o nível de umidade a partir da resistência medida, um reservatório de água, uma bomba de sucção.
Mais >>
Visão Geral do Projeto SIAT
Sensores
Umidade
Temperatura
M.Controlador
Interface Potência para bomba d'água
Interface para sensores
Bomba d'água
SIAT - Irrigação Automatizada
Documentação
Solo + Planta
Software
Firmware
Banco de Dados
Aplicativo web
Aplicativo desktop
Vantagens da Irrigação Automatizada
Economia de mão de obra
Possibilita irrigações noturnas sem necessidade de acompanhamento
Economia de tempo
Diminuição do custo de bombeamento

O projeto, apesar de apresentar uma simplicidade na implementação, é complexo em sua programação e de dificil montagem.
Deve-se ressaltar que houveram problemas na placa de circuito impresso pois a solda estava quebrada, além de se ter soldado a alimentação do circuito com a polaridade inversa, o que acarretou na inversão do sensor de umidade e também havia curtos circuitos.
Entretanto o resultado, mostrado abaixo, foi satisfatório e a montagem do projeto foi executada com êxito.





8 Conclusões

Com o projeto finalizado tivemos um resultado bastante satisfatório e foi possível obter uma maior compreensão dos elementos envolvidos. Percebemos que a utilidade do projeto não se resume em apenas um objetivo, mas sim com uma complexidade maior que o esperado.
Entretando os materiais utilizados não foram exatamente precisos, pois para uma implementação real no mercado industrial seria necessário equipamentos que apresentassem maior resistência e qualidade.
Para um simples projeto como esse, notamos que varias modificações devem ser feitas nas variáveis para que chegassemos no resultado final.
Ressalta-se que foi de grande importância a interação entre os membros do grupo para o sucesso do projeto, mostrando assim a relevância de um trabalho em equipe.

9 Referências Bibilográficas

- Kompozer;
- Geany;
- Firebug;
- Filezilla;
- Writelatex;
- Redmine;
- Eagle;
- Isopro;
- Mindomo.
- http://zilocchi.com.br/modulos-didaticos/video/modulo-de-eletronica-analogica-basico-modelo-mpla-1201-7.html
- https://www.shoppingdolaboratorio.com.br/produto.php?cod-produto=1040224
- http://www.eletrodex.com.br/microcontrolador-pic18f2550-i-so-smd.html

10 Apêndices

10.1 Códigos fonte do projeto

Código fonte do Arduino

```
/*
Controle de irrigaçao

12/11/2014

A0 -> sensor de umidade: 5V seco, 0V encharcado
A1 -> sensor nivel agua: > 4V seco, senao com agua

13 -> acionamento bomba dagua
*/

#include<stdlib.h>

int bomba = 13;
float umidade, nivel_agua;
float sp_umidade, sp_nivel_agua;

char buff [10];


void setup() {
  Serial.begin(9600);
  pinMode(bomba, OUTPUT);

  sp_umidade = 40;
  sp_nivel_agua = 50;
}

void loop() {;

  umidade = analogRead(A0);
  nivel_agua = analogRead(A1);

  umidade = map(umidade, 0, 1023, 100, 0);
  nivel_agua = map(nivel_agua, 0, 1023, 100, 0);

  Serial.print("Um:");
  dtostrf(umidade,6,2,buff);
  Serial.println(buff);
  Serial.print("Na:");
  dtostrf(nivel_agua,6,2,buff);
  Serial.println(buff);

  if (nivel_agua > sp_nivel_agua){
    if(umidade < sp_umidade){
```

```
      digitalWrite(bomba, HIGH);
    } else{
      digitalWrite(bomba, LOW);
    }
  } else {
      digitalWrite(bomba, LOW);
      Serial.print("Na:");
      dtostrf(nivel_agua,6,2,buff);
      Serial.println(buff);
      Serial.println("VAZIO");
  }
  delay(1000);
  }
```

Código fonte do Aplicativo

```
#! /usr/bin/env python
# -*- coding: iso-8859-1 -*-

import os
import commands
import serial
import time
from threading import Thread
from datetime import datetime

import sys
import MySQLdb

from gi.repository import Gtk

porta = '/dev/ttyACM0'
baud_rate = 9600
Obj_porta = None
THRD = None

conSql = MySQLdb.connect('localhost', 'root', 'aluno') # (servidor, usuario, senha)
conSql.select_db('irrigacao') # seleciona o banco de dados na conexao
curSql = conSql.cursor() # cria o cursor para o BD

'''
cursor.execute('ALGUM SQL') # faz alguma query sql
# aqui você pode usar três maneira para pegar o resultado:
rs = cursor.fetchone() # pega uma linha;
rs = cursor.fetchall() # pega todas as linhas;
rs = cursor.dictfetchall() # pega todas as linhas, cada linha tem um dicionário com os nom
print(rs[0]) # imprime o valor da primeira coluna
'''

class IGraf(object):
```

```
def __init__(self):

builder = Gtk.Builder() #Criamos uma instancia da classe builder do Gtk
builder.add_from_file("irrigacao.glade") #Carregamos o arquivo do Glade

#Obtemos o widget window1 (nossa janela principal)
self.window = builder.get_object("window1")

#Obtemos o widget text_view (a area de texto do nosso programa)
self.txt_recebido = builder.get_object("txt_recebido")

#Obtemos o widget text_entry (a linha de texto do nosso programa)
self.txt_enviar = builder.get_object("txt_enviar")

#Obtemos o widget ComboBox
self.cb_portas = builder.get_object("cb_portas")

#Exibimos a janela do programa
self.window.show()

#Conectamos os sinais que definimos para cada widget no Glade
builder.connect_signals({"on_bt_conectar_clicked" : self.on_bt_conectar_clicked,
"on_bt_desconectar_clicked" : self.on_bt_desconectar_clicked,
"on_bt_enviar_clicked" : self.on_bt_enviar_clicked,
"on_bt_sair_clicked" : self.on_bt_sair_clicked })


def on_bt_sair_clicked(self, widget):
self.on_bt_desconectar_clicked(self)
Gtk.main_quit()

def on_bt_conectar_clicked(self, widget):
#print "Conectar"
global Obj_porta
porta = "/dev/ttyACM0"
txt = self.txt_recebido.get_buffer()
if porta <> None:
txt.insert_at_cursor("Conectado!\n")
Obj_porta = serial.Serial(porta, baud_rate)
global THRD
THRD = Thread(target=self.thrd_loop)
THRD.morta=False
THRD.start()
else:
txt.insert_at_cursor("Selecione uma porta!\n")

def on_bt_desconectar_clicked(self, widget):
txt = self.txt_recebido.get_buffer()
txt.insert_at_cursor("Desconectado!\n")
if THRD != None:
```

```
THRD.morta = True

def on_bt_enviar_clicked(self, widget):
lin = self.txt_enviar.get_text()
txt = self.txt_recebido.get_buffer()
if txt.get_line_count() > 15:
txt.delete(txt.get_iter_at_line(0),txt.get_iter_at_line(1))
txt.insert_at_cursor("Enviado: ")
txt.insert_at_cursor(lin)
txt.insert_at_cursor('\n')
Obj_porta.write(lin)

def verifica_portas(self):
self.cb_portas.remove_all()
portas = commands.getoutput("ls /dev/ttyACM*") # lista portas do arqduino
portas_ativas = portas.split()
for numero, porta in enumerate(portas_ativas):
self.cb_portas.append_text(porta)

def le_porta(self): # lê a porta seria, mostra na tela e envia para o BD
valor = Obj_porta.readline()
txt = self.txt_recebido.get_buffer()
txt.insert_at_cursor(valor)
if txt.get_line_count() > 15:
txt.delete(txt.get_iter_at_line(0),txt.get_iter_at_line(1))
self.grava_bd(valor)


def thrd_loop(self): # funcao a ser executada pela thread
while not THRD.morta:
self.le_porta()
time.sleep(0.5)

def grava_bd(self,linha):
linha = linha[:-2] # para remover \r\n do final da linha
hoje = datetime.now()
data =  "%s-%s-%s" % (hoje.year,hoje.month,hoje.day)
hora = "%s:%s:%s" % (hoje.hour,hoje.minute,hoje.second)
print data
print hora
qry = "Insert into dados (id,data,hora,nivel,umidade,bomba) values (0,'"+data+"','"+hora+"
#qry += "".join(linha)
print qry
curSql.execute(qry)
conSql.commit()

if __name__ == "__main__":
app = IGraf()
Gtk.main()
```

10.2 Datasheet de componentes

10.2.1 Módulo de Eletrônica Analógica Básico modelo MPLA-1201

Figura 1: Módulo de Eletrônica Analógica Básico modelo MPLA-1201

O Módulo Prático Laboratorial de Eletrônica Analógica Básico MPL-A1201 é um equipamento desenvolvido para auxiliar na montagem e teste de circuitos de eletrônica básica e circuitos analógicos que vão desde lei de Ohm até aplicações com amplificadores operacionais, passando pelos principais circuitos discretos e com semicondutores (amplificadores, diodos, retificadores, fontes lineares, etc.).
É projetado para um menor espaço na bancada, dispõe de importantes recursos didáticos como 03 protoboard (matrizes de contatos) para as montagens, testador de continuidade, fontes DC variáveis positiva e negativa protegidas, indicação luminosa de curto-circuito, fonte AC e entre outras funcionalidades.Integrado em um único equipamento o Módulo MPL-A1201 possui as principais características:
- Bastidor horizontal em aço com pintura eletrostática a pó (anti-ferrugem);
- Fonte AC com transformador de tape central;
- Fonte de alimentação DC com ajuste variável positiva através de potenciômetro com saída de até +24V e totalmente protegida contra curto-circuito;
- Fonte de alimentação DC com ajuste variável negativa através de potenciômetro com saída de até -24V e totalmente protegida contra curto-circuito;
- A saída das fontes variáveis (positiva e negativa) possuem proteções independentes contra curto-circuito, caso na montagem das experiências ocorrer um curto entre as fontes imediatamente há o corte da tensão e corrente e indicação luminosa através de led indicando o curto.
- Conectores de acesso abertura por mola, padrão industrial, não entope, não desgasta e suporta fios de 14 até 22 AWG e borne para entrada de cabo banana de até 4mm.
- Testador de continuidade com indicação luminosa e sinal sonoro (acompanha um par de ponta de prova).
- Componentes protegidos, localizados na parte interna do equipamento, dificultando a queima acidentalmente ou propositalmente do equipamento, acesso somente com a abertura do bastidor;
- 03 matrizes de contatos (protoboard's) de 550 pontos cada totalizando 1650 pontos para montagem de circuitos eletrônicos.
- Estojo com componentes eletrônicos para realização de experiências de

Eletrônica Básica e Eletricidade Básica.
- Manual de operação e experiências;
- 01 ano de garantia.

10.2.2 Multímetro digital (ICEL MD-6450) True RMS, RS-232C

A figura 2 apresenta o multímetro utilizado.

Figura 2: Multímetro digital (ICEL MD-6450) True RMS, RS-232C

- Visor: De cristal líquido (“LCD”), 4000 dígitos multifuncional com iluminação.
- Funções: tensão contínua e alternada, corrente contínua e alternada, resistência, teste de continuidade, teste de diodo, temperatura, freqüência, capacitância e ciclo de atividade.
- Polaridade: Automática. O sinal negativo (–) será exibido automaticamente.
- Indicação de sobrecarga: O símbolo de "OL"será exibido no visor.
- Indicação de bateria gasta: O visor exibirá o símbolo de uma pilha quando restar aproximadamente 10 por cento da energia útil da bateria.
- Temperatura de operação: De 0º a 40ºC.
- Umidade de operação: Menor que 75 por cento sem condensação.
- Temperatura de armazenagem: De –10º a 50ºC (<80 por cento RH sem condensação).
- Alimentação: Uma bateria de 9V.
- Taxa de amostragem: três vezes por segundo.
- Proteção: Dois fusíveis de cerâmica, sendo um de 10A/600V e outro de 0,5A/600V.
- Dimensões: 177x85x40mm.
- Peso: 300g (incluindo a bateria).
- O multímetro vem acompanhado de um manual de instruções, um cabo RS-232C, um jogo de pontas de prova, um jogo de garras jacaré, um termopar TP-01 e um CD-rom com software para sistema operacional Windows 95/98/2000/Me/XP.
- Altitude máxima de operação: 2.000 metros
- O multímetro obedece às normas IEC1010, CAT II -1000V e CAT III – 600V.

10.2.3 Microcontrolador PIC18F2550-I/SO SMD

A figura 3 apresenta o PIC utilizado.

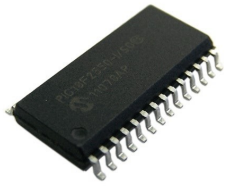


Figura 3: Microcontrolador PIC18F2550-I/SO SMD

Fabricante: Microchip
Program Memory (Flash): 16384 Words (32K Bytes)
Data Memory (SRAM): 2048 bytes
Data Memory (EEPROM): 256 bytes
I/O: 24
10 bit A/D: 10 canais
CCP (PWM): 2
EUSART: 1
SPI: Sim
Comparators: 2
Timers 8 bits: 1
Timers 16 bits: 3
Encapsulamento: I/SO
Número de Pinos: 28
Montagem SMD: Sim
Temperatura de funcionamento contínuo MIN: -40 °C a 85 °C
VDD range: 4.2V a 5V.

10.2.4 Capacitor Suntan 2G224K

Poliéster metalizado capacitor de filme
Especificações:
Temperatura de Operação -40 a + 85 graus
Tensão DC 100V, 250V, 400V, 630V
Faixa de capacitância de 0,01 a 6.8 nanoF
Fator de dissipação: menor ou igual a 0.01 (em 1KHz)

10.2.5 Eletrobomba de Lavador

- Bomba gasolina
- Partida á frio
- 1 saída 12V - Universal
- Código TSA : 820001G
- Código Original: 500.14.364

10.2.6 Arduino Uno

A figura 4 apresenta o Arduino utilizado. microcontrolador ATmega328;

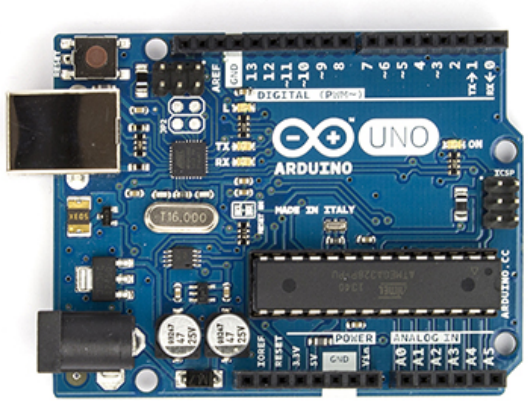


Figura 4: Arduino Uno R3 Frente

Tensão operacional de 5V;
Tensão de entrada (recomendada) 7-12V;
Tensão de entrada (limites) 6-20V;
Digital I / O pinos 14 (dos quais 6 oferecem saída PWM);
Pinos de entrada analógica 6;
Corrente DC por I / O Pin 40 mA;
Corrente DC para 3.3V Pin 50 mA;
Memória Flash 32 KB (ATmega328), dos quais 0,5 KB usados pelo bootloader SRAM 2 KB (ATmega328);
EEPROM 1 KB (ATmega328);
Velocidade de clock de 16 MHz.